Review: What problems have we been working to solve:

- Ignorance. Want to Understand things (Mathematics)
- Computation is good, but not satisfying alone with knowing Why. (Need Proof)
- In the middle of a proof, we need to use other facts that themselves need to be established; infinite (or circular) recursion (i.e. A is true because of B. And B is true because of C. And C is true because of A.) Solution: Axiomatic method. We will hold truths to be self-evident. Better: we're declaring the rules of the game.
- Since words are described in terms of other words, we will again face infinite (or cyclic) recursion if we try to give definitions to the most basic objects. Solution: Undefined Terms.
- As we try to build up our mathematical theory, we may face False Positives (proof that "look" OK turn out not to be sufficiently justified, e.g., Prop I.1 that circles intersect) and False Negatives (proving something that looks Wrong but actually isn't, there's no contradiction, e.g., in hyperbolic geometry, AAA implies congruence). Moreover, debate about which statements should be Postulates/Axioms vs Theorems (e.g. Parallel / 5th Postulate) Solution: Work completely formally!!

Where did we end up:
Opened the library and got undefined `Types`

```
import Mathlib -- importing the math library of Lean4

class EuclideanPlane where
  Pt : Type -- undefined object
  Line : Type -- undefined!!!
  Circ : Type -- undefined
```

We stated a definition of equilateral triangle:

```
def IsEquilateralTriangle (a b c : Pt) : Prop := (dist a b = dist b c) ∧
(dist a b = dist a c)
```

We filled in some more axioms:

```
  Pt_on_Line : Pt → Line → Prop
  -- Postule 1: (Note: "∀" = "for all", "∃" = "there exists", "∧" = "and")
  Line_of_Pts : ∀ a b : Pt, ∃ L : Line, (Pt_on_Line a L) ∧ (Pt_on_Line b L)
  -- Postulate 2 is not needed, all our "Lines" are already infinite
```

```
  Pt_on_Circ : Pt → Circ → Prop
  Center_of_Circ : Pt → Circ → Prop
  -- Postulate 3:
  Circ_of_Pts : ∀ a b : Pt, ∃ α : Circ, (Center_of_Circ a α) ∧ (Pt_on_Circ b
α)
  dist : Pt → Pt → ℝ
  -- `Circs_inter` "means:" generic, not tangential intersection
  Circs_inter : Circ → Circ → Prop
  Pts_of_Circs_inter : ∀ α β : Circ, Circs_inter α β → ∃ a b : Pt,
    Pt_on_Circ a α ∧ Pt_on_Circ b α ∧ Pt_on_Circ a β ∧ Pt_on_Circ b β ∧ a ≠
b
```

Then we started the proof of Prop I.1:

```
theorem PropIp1 (a b : Pt) : ∃ (c : Pt), IsEquilateralTriangle a b c := by
-- " := by " that's the beginning of the proof
/-
What is the current goal state?? (" ⊢ " = "goal")
a b : Pt
⊢ ∃ (c : Pt), IsEquilateralTriangle a b c
-/
-- Step 1: Create a circle centered at `a` with `b` on the circle (using
Postulate 3)
-- Let's have the fact that `Circ_of_Pts`:
have := Circ_of_Pts a b
/-
Goal state:
a b : Pt
this : ∃ α, Center_of_Circ a α ∧ Pt_on_Circ b α
⊢ ∃ c, IsEquilateralTriangle a b c
-/
-- need to obtain the circle and two properties
obtain ⟨α, a_center_of_α, b_on_α⟩ := this
/-
New goal state:
a b : Pt
α : Circ
a_center_of_α : Center_of_Circ a α
b_on_α : Pt_on_Circ b α
⊢ ∃ c, IsEquilateralTriangle a b c
```

```
-/
-- Step 2: Draw circle with center `b` passing through `a`
have Vaibhav := Circ_of_Pts b a
obtain ⟨β, b_center_of_β, a_on_β⟩ := Vaibhav
```

Homework: Click the link: https://codespaces.new/leanprover-community/mathlib4 and just see if you can get this software running in your browser.

Now let's talk about your previous homework: When do two circles in the Euclidean plane actually intersect? What is a good characterization of that phenomenon?

- Idea 1: Let R_1 be the radius of one circle, R_2 of the other. Then $|R_1 - R_2| < dist(centers) < R_1 + R_2$.
- Proposed Predicate: Given circles α and β, if there exists a point a which is ON α and INSIDE β AND (There exists a point b which is ON β and INSIDE α), then the circles to intersect.

Following this idea, let's add a few axioms:

Modus Ponens: ((P → Q) AND P) → Q

Khalil's question: Why → (arrow) and not ∧ (AND) symbols?

- If you're trying to prove P ∧ Q → R. This is logically equivalent to : P → Q → R (Star problem: Why?)
- But if you're trying to prove that P → Q ∧ R, then you can't turn ∧ into → (Those are not logically equivalent)

At the end of Lecture5, we were up to this:

```
import Mathlib -- importing the math library of Lean4

class EuclideanPlane where
  Pt : Type -- undefined object
  Line : Type -- undefined!!!
  Circ : Type -- undefined
  Pt_on_Line : Pt → Line → Prop
  -- Postule 1: (Note: "∀" = "for all", "∃" = "there exists", "∧" = "and")
  Line_of_Pts : ∀ a b : Pt, ∃ L : Line, (Pt_on_Line a L) ∧ (Pt_on_Line b L)
  -- Postulate 2 is not needed, all our "Lines" are already infinite
  Pt_on_Circ : Pt → Circ → Prop
  Center_of_Circ : Pt → Circ → Prop
```

```
  -- Postulate 3:
  Circ_of_Pts : ∀ a b : Pt, ∃ α : Circ, (Center_of_Circ a α) ∧ (Pt_on_Circ b
α)
  dist : Pt → Pt → ℝ
  Circs_inter : Circ → Circ → Prop -- "meaning:" generic, not tangential
intersection
  Pts_of_Circs_inter : ∀ α β : Circ, Circs_inter α β → ∃ a b : Pt,
    Pt_on_Circ a α ∧ Pt_on_Circ b α ∧ Pt_on_Circ a β ∧ Pt_on_Circ b β ∧ a ≠
b
  Pt_in_Circ : Pt → Circ → Prop
  Circs_inter_iff : ∀ (α β : Circ), Circs_inter α β ↔
    (∃ (a b : Pt), Pt_on_Circ a α ∧ Pt_in_Circ a β ∧
      Pt_on_Circ b β ∧ Pt_in_Circ b α)
  -- Now we define a circle (!!!):
  Pt_on_Circ_iff : ∀ (a b c : Pt) (α : Circ), Center_of_Circ a α →
    Pt_on_Circ b α → (Pt_on_Circ c α ↔ dist a c = dist a b)
  Pt_in_Circ_iff : ∀ (a b c : Pt) (α : Circ), Center_of_Circ a α →
    Pt_on_Circ b α → (Pt_in_Circ c α ↔ dist a c < dist a b)

variable [EuclideanPlane]

namespace EuclideanPlane

-- lets try to state and prove Proposition I.1
-- first we need to define what it means for a triangle to be equilateral
-- Def: Given three points `a`, `b`, and `c`, they are the corners of an
equilateral triangle
-- if: dist from `a` to `b` = dist from `b` to `c` AND dist from `a` to `b`
= dist `a` to `c`
def IsEquilateralTriangle (a b c : Pt) : Prop := (dist a b = dist b c) ∧
(dist a b = dist a c)

-- Statement of Prop I.1: Given two points `a` and `b`, there exists a point
`c` so that
-- `a`, `b`, and `c` form an equilateral triangle
theorem PropIp1 (a b : Pt) : ∃ (c : Pt), IsEquilateralTriangle a b c := by
-- " := by " that's the beginning of the proof
/-
What is the current goal state?? (" ⊢ " = "goal")
 a b : Pt
```

```
 ⊢ ∃ (c : Pt), IsEquilateralTriangle a b c
-/
-- Step 1: Create a circle centered at `a` with `b` on the circle (using
Postulate 3)
-- Let's have the fact that `Circ_of_Pts`:
  have Step1 : ∃ (α : Circ), Center_of_Circ a α ∧ Pt_on_Circ b α :=
    Circ_of_Pts a b
/-
Goal state:
a b : Pt
this : ∃ α, Center_of_Circ a α ∧ Pt_on_Circ b α
⊢ ∃ c, IsEquilateralTriangle a b c
-/

  -- need to obtain the circle and two properties
  obtain ⟨α, a_center_of_α, b_on_α⟩ := Step1
/-
New goal state:
a b : Pt
α : Circ
a_center_of_α : Center_of_Circ a α
b_on_α : Pt_on_Circ b α
⊢ ∃ c, IsEquilateralTriangle a b c
-/
-- Step 2: Draw circle with center `b` passing through `a`
  have Step2 : ∃ (β : Circ), Center_of_Circ b β ∧ Pt_on_Circ a β :=
    Circ_of_Pts b a
  obtain ⟨β, b_center_of_β, a_on_β⟩ := Step2

  -- Next step is to claim that the circles do intersect
  have Step3 : Circs_inter α β := by
    -- I don't want to prove this right now; let's see if we can
    -- complete the proof, postponing this part. That's called `sorry`
    sorry

  have Step4 : ∃ (c d : Pt), Pt_on_Circ c α ∧ Pt_on_Circ d α ∧
    Pt_on_Circ c β ∧ Pt_on_Circ d β ∧ c ≠ d :=
      Pts_of_Circs_inter α β Step3

  -- homework : what is the next formal line in the proof, how do we
```

```
  -- get the points `c` and `d` out of Step4?? (Hint: Look above)

  -- Recall syntax: `have` FactName `:` Fact to prove `:=` Proof
```